#include <iostream>

using namespace std;

struct CircArrayQueue

{

//core variables

int back;

int front;

int length;

int\* Array;

//member functions

int peek();

int size();

void enQueue(int value);

int deQueue();

//Let's make a constructor. This will initialize our main array.

CircArrayQueue(int size = 10)

{

int number = 10;

while (size > number) {

number \*= 2;

}

front = back = -1;

length = size;

Array = new int[number];

}

//Time for a destructor

~CircArrayQueue() {

delete[] Array;

}

CircArrayQueue& operator =(const CircArrayQueue& newArray) {

for (int i = 0; i < length; i++) {

newArray[i] = Array[i];

}

}

};

//First, I did the size function because it was easy. It just returns length.

int CircArrayQueue::size()

{

return length;

}

//enqueue adds a value to the "queue"

void CircArrayQueue::enQueue(int value)

{

//check if the array is full here by using the mod function. This works because the array is circular, so even if front is double of length, it still works.

if ((back == (front - 1) % (length - 1)) or (front == 0 and back == length - 1))

{

CircArrayQueue NewArray(length \* 2);

}

else if (front == -1)

{

front = 0;

back = 0;

Array[back] = value;

}

else if (back == length - 1 and front != 0)

{

back = 0;

Array[back] = value;

}

else

{

back++;

Array[back] = value;

}

}

int CircArrayQueue::peek()

{

if (front == -1)

{

return NULL;

}

return Array[front];

}

int CircArrayQueue::deQueue()

{

if (front == -1)

{

return NULL;

}

int data = Array[front];

Array[front] = -1;

if (front == back)

{

front = -1;

back = -1;

}

else if (front == length - 1)

front = 0;

else

front++;

return data;

}

int main()

{

CircArrayQueue q(100);

for(int i = 0; i < 161; i++)

q.enQueue(20);

cout << q.size();

return 0;

}